Capstone Project Report

Marcel Stolin
marcelstolin@gmail.com

1 Definition

1.1 Project Overview

Image classification is an active research field in multimedia. Applications of image classification are widely used in security, healthcare, entertainment and many more to face real world problems. The goal of this project is to develop an image classifier to predict dog breeds and it is mainly created for entertainment purposes. However, serious research projects about animal classification exists. For example Trnovszky, Kamencay, Orjesek, Benco and Sykora published a paper where they created a Convolutional Neural Network (CNN) to propose the animal species from an images of an animal [7].
This projects is also one of the options, given by Udacity, for a capstone project. It comes with a predefined problem statement and a Jupyter Notebook to complete. Udacity also provides the data of dog images and human images. The dataset for dog images is a custom dataset from Udacity. The images of human faces come from the LFW (Labeled Face in the Wild) [2] dataset from the University of Massachusetts.

1.2 Problem Statement

As mentioned in section 1.1 this project comes with a predefined problem statement and a Jupyter Notebook. The main goal is to create a CNN to predict a dog breed from a given image. If the image only shows a human, then the algorithm will detect the human and return the dog breed for the human face. To achieve the project goal, a CNN has to be made from scratch or a pretrained model has to be used. To create a CNN from scratch and compare the performance to a pretrained model is a mandatory task in this project.

According to this information and to the given Jupyter Notebook, the project can be cur down to the following ordered list of problems:

1. Create a human face detector
2. Create a dog detector
3. Create a CNN to predict the dog breed

4. Create an algorithm, which will return the predicted dog breed for the image

The listed points are just a simple overview for the whole project. The given data also needs to be imported and processed, which is described in detail in section 3.1.

The algorithm comes with it own requirements:

1. The image shows a dog ⇒ Return the predicted dog breed
2. The image shows a human ⇒ Return the predicted dog breed with a different text
3. The image shows neither man nor human ⇒ Print an error message

1.3 Metrics

To evaluate a classification model, it is common to use the Accuracy, Precision, Recall and F1-Score metric. A Confusion Matrix can be used to visualize the results of a classifier.

Confusion Matrix The Confusion Matrix makes it it easy to see if a model confuses a class. All above mentioned metrics are based on a Confusion Matrix.

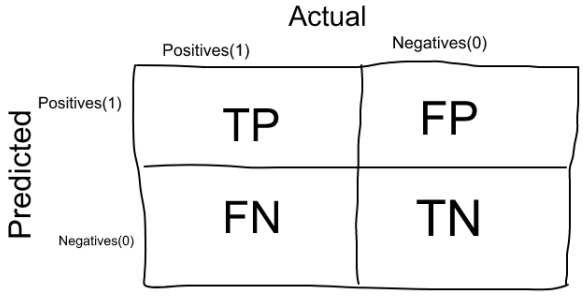

Figure 1: Confusion Matrix [3].

- **True Positives (TP)** are cases when the actual class and the predicted are True (1)
- **False Positives (FP)** are cases when the actual class is False (0) and the predicted is True (1)
- **True Negatives (TN)** are cases when the actual class and the predicted class are False (True)

- **False Negatives (FN)** are cases when the actual class is True (1) and the predicted is False (0)

Accuracy Accuracy is the number of correct predictions made over all predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

Precision Precision is measure which tells us, what proportion of predicted positives are truly positives.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall The recall metric tells us, what proportion of actual positives are correctly classified.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

F1-Score The F1-Score is the harmonic mean of precision and recall.

$$F1Score = 2 * Precision * \frac{Recall}{Precision + Recall} \tag{4}$$

2 Analysis

2.1 Data Exploration

The dog images dataset has already been split into test, train, and valid sets. In total the dataset contains 8351 image files of 133 unique dog breeds. The images of the Udacity dog dataset have different dimensions, therefore the images have to be resized, which is described in detail in section 3.1. The LFW dataset contains 13234 images of 5749 unique individuals. Images in the LFW dataset have a dimension of 250x250 pixels.

Figure 2: Example images from the datasets. The first images shows the Akita breed, the second image shows the actor Aaron Eckhart.

2.2 Exploratory Visualization

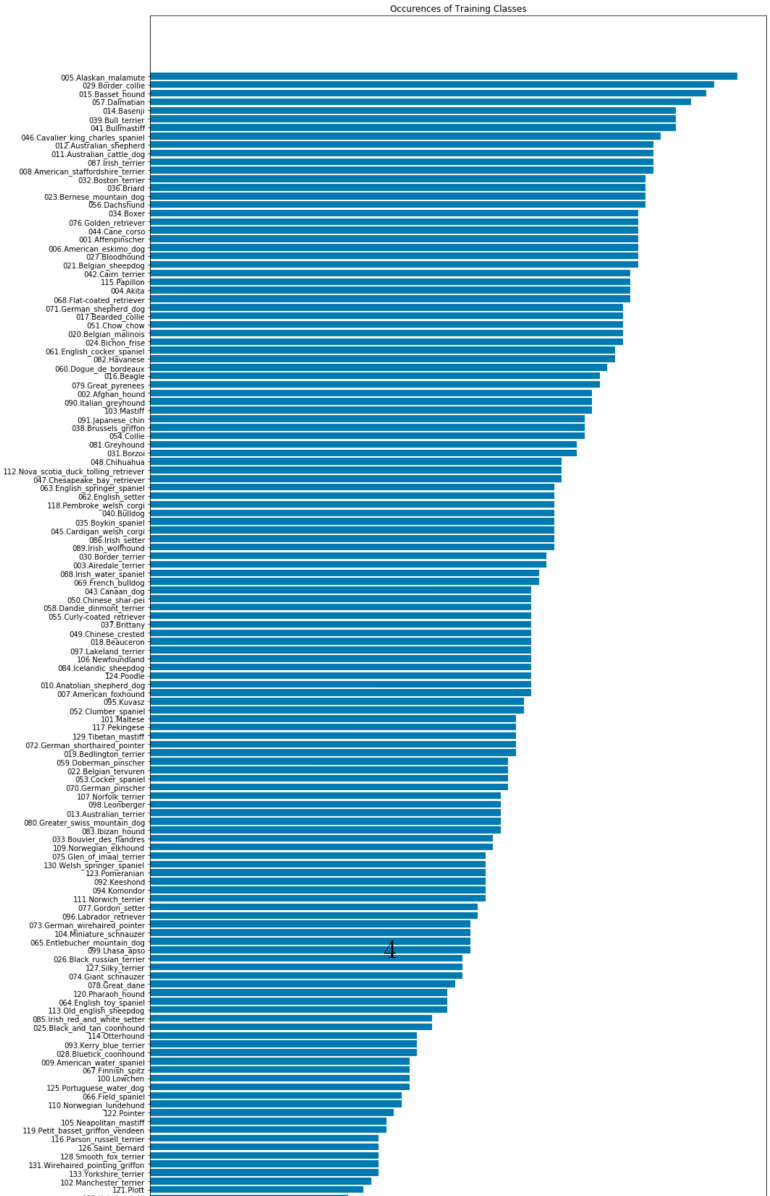

Figure 3 shows the occurences of each individual dog breed in the train dataset. As the bar shows, the breeds are not evenly distributed. Therefore the CNNs of this project, will predict dog breeds with a higher occurence (e.g. Alaskan Malamute), better than breeds with a lower occurence (e.g. Norwegian Buhund).

2.3 Algorithms and Techniques

Face Detection One part of the problem is to detect humans. Therefore a detector for human faces is needed. OpenCV provides the Haar Cascade classifier [8]. The classifier is trained from images with an machine learning approach based on a cascade function, to detect objects in images [1]. OpenVC provides three different pretrained models to detect human faces: FrontalFaceAlt, FrontalFaceAlt2, FrontalFaceDefault. The comparison result showed, that FrontalFaceAlt is the best pretrained model for our use case. The model has an accuracy of 100% for human faces.

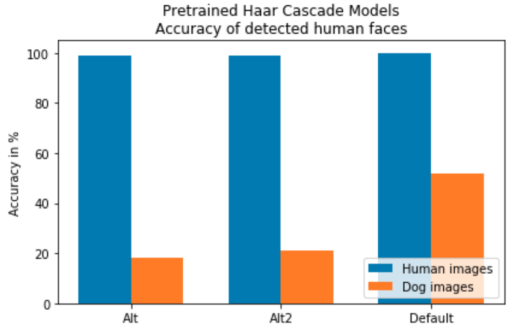

Figure 4: Accuracy comparison of all three Haar cascade models.

Dog Detection VGG-16 [6] is a useful pretrained network to detect dogs. The VGG-16 network is recommended by the creators of the provided notebook. It is also possible to use other pretrained networks like Inception-v3 or ResNet-50 [5]. I decided to focus on VGG-16 for the dog detection, because in Transfer Learning, the ResNet-50 network will be used anyway.

```
VGG16
-----
Accuracy of detected dogs in dog images: 97.0%
Accuracy of detected dogs in human images: 0.0%
```

Figure 5: Output of the VGG-16 model accuracy on the training datasets.

Dog Classifier This project has two mandatory tasks: Create a CNN from scratch and create a CNN using Transfer Learning. To create a CNN from

scratch, the framework provided by the notebook is PyTorch. PyTorch is a machine learning library developed by facebook [4].
For Transfer Learning I have decided to use the ResNet-50 pretrained network. The ResNet-50 network is trained with 152 layers on the ImageNet dataset and has won the 1st price on the ILSVRC 2015 classification task [5]. Therefore, the ResNet-50 is the best fit for the problem of this project.

2.4 Benchmark

The benchmark requirements of this project are: The CNN made from scratch has to have an accuracy greater than 10% and the Transfer Learning CNN has to have an accuracy greater than 60%. The requirements are mandatory tasks from Udacity.
There are no requirements about the dog detector or the human face detector. I have decided that an accuracy of at least 90% has to be reached for these models.

3 Methodology

3.1 Data Preprocessing

The ResNet-50 network is trained with images of a 224x224 dimension. Therefore I decided to use that image size for the network I have to create from scratch. I will also use the same values for image normalization parameters like torchvision. The torchvision normalization parameters are:

- **mean:** [0.485, 0.456, 0.406]
- **std:** [0.229, 0.224, 0.225]

According to these parameters, all input images will be preprocessed like the following:

1. Resize to 224x224 with center crop
2. Normalize with torchvision mean/std values

The training data will be augmented, because it will increase the diversity of the test data, without adding new images and prevents overfitting. I decided to augment the images with a random horizontal flip and random rotation. Only horizontal flipping is activated, because dogs cant be upside down. The training images will be preprocessed like the following:

1. Randomly resize the image to 224x224 with center crop
2. Random horizontal flip
3. Random rotation of 10 Degrees
4. Normalize with torchvision mean/std values

3.2 Implementation

Split the Dataset The first task was to split the datasets into train, valid and test data. The DataLoader class from PyTorch is a good solution to handle this case. As mentioned in section 2.1, the dataset has already been split into train, valid and test, therefore it was easy to create three custom DataLoader and load the specific files. How the images get preprocessed is explained in section 3.1.

Create a CNN from scratch The second task was to create a CNN from scratch. To accomplish that, the notebook provides the machine learning library PyTorch to handle this task.
The CNN has three convolutional layers, the first two layers have a stride of 2 to downsize the input size by two. After each convolutional layer, the output will be normalized by a batch normalization layer. After that, the ReLU activation function will be applied to the output. A max. pooling layer will downsize the output of each convolutional layer by two, after batch normalization and ReLU activation. To prevent overfitting the CNN has a dropout layer, the probability that an element will be zero-ed is 0.3. To create the output of the CNN, two fully connected linear layers will produce the 133-dim output.
For training, the Cross Entropy loss function and the Stochastic Gradient Descent (SGD) criterion have been used.

```
Net(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (dropout): Dropout(p=0.3, inplace=False)
  (conv_bn1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv_bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (conv_bn3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (fc1): Linear(in_features=6272, out_features=512, bias=True)
  (fc2): Linear(in_features=512, out_features=133, bias=True)
)
```

Figure 6: CNN architecture.

Transfer learning CNN For the transfer learning implementation I have decided to use the ResNet-50 network, as mentioned in section 2.3. ResNet-50 is a pretrained network, so no architecture has to be defined by myself. The only thing that was necessary to complete the task, was to train the network with our train dataset. For training, the same loss function and criterion have been used.

3.3 Refinement

To come up with a good number of epochs, I updated the train function to save and return a history of the loss and validation accuracy metric. At first, I started to train the CNN with 50 epochs. The visualization makes clear, that after the 40th epoch, no significant improvement has been made. The validation accuracy of the model will go up and down in range of 0.25 and 0.28. Also the validation loss stays between 2.9 and 3.0. Therefore I have decided to train the CNN with 40 epochs to prevent overfitting.

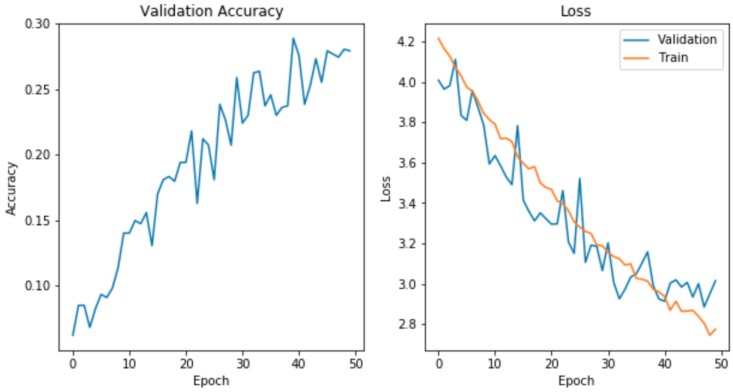

Figure 7: Accuracy and loss history of the CNN training.

4 Results

4.1 Model Evaluation and Validation

The model will be trained for several epochs. In each epoch the trained model gets validated against the validation dataset. The model with the lowest validation loss value will be saved. After training, the accuracy for the saved model gets determined. For testing the test dataset will be used.
During this project, two networks will be trained, a network made from scratch and a pretrained network for transfer learning.

4.1.1 Scratch CNN

The network I had to create from scratch has a final accuracy of 27% and a loss value of 2.96. Therefore the model hits the benchmark requirements.
An accuracy of 27% is not a good result for an image classification model. The classification report makes clear that the network is not able to detect 31 breeds out of 133. All of these 33 breeds have a precision, recall and F1-score value of 0. Because of these results, I decided to use a pretrained network for the transfer learning task.

	precision	recall	f1-score
001.Affenpinscher	0.23	0.62	0.33
002.Afghan_hound	0.15	0.25	0.19
003.Airedale_terrier	0.00	0.00	0.00
004.Akita	0.10	0.12	0.11
005.Alaskan_malamute	0.33	0.40	0.36
006.American_eskimo_dog	0.21	0.62	0.31
007.American_foxhound	0.00	0.00	0.00
008.American_staffordshire_terrier	1.00	0.12	0.22
009.American_water_spaniel	0.50	0.25	0.33
010.Anatolian_shepherd_dog	0.00	0.00	0.00
011.Australian_cattle_dog	0.50	0.11	0.18
012.Australian_shepherd	0.50	0.44	0.47
013.Australian_terrier	0.33	0.17	0.22
014.Basenji	0.14	0.22	0.17
015.Basset_hound	0.50	0.30	0.37

Figure 8: Classification metrics results of the scratch CNN for the first fifteen dog breeds.

4.1.2 Transfer Learning Pretrained CNN

For transfer learning I have decided to use the ResNet-50 network, this is described in section 2.3. The pretrained model has reached a test accuracy of 81% and a test loss value of 1.02. Therefore the model reaches the benchmark requirements.
The model shows significant improvements in the classification report. There is no breed with a precision, recall or F1-score value of 0.

	precision	recall	f1-score
001.Affenpinscher	1.00	1.00	1.00
002.Afghan_hound	0.89	1.00	0.94
003.Airedale_terrier	0.75	1.00	0.86
004.Akita	1.00	0.62	0.77
005.Alaskan_malamute	0.77	1.00	0.87
006.American_eskimo_dog	1.00	1.00	1.00
007.American_foxhound	1.00	0.57	0.73
008.American_staffordshire_terrier	1.00	1.00	1.00
009.American_water_spaniel	1.00	0.25	0.40
010.Anatolian_shepherd_dog	0.83	0.83	0.83
011.Australian_cattle_dog	0.86	0.67	0.75
012.Australian_shepherd	1.00	0.89	0.94
013.Australian_terrier	0.86	1.00	0.92
014.Basenji	0.69	1.00	0.82
015.Basset_hound	0.91	1.00	0.95

Figure 9: Classification metrics results of the ResNet-50 for the first fifteen dog breeds.

4.2 Justification

Both models exceeded the benchmarks and have therefore fulfill the requirements. The ResNet-50 network has an improvement of 54% from the scratch CNN. The classification model makes clear, that the ResNet-50 outperformed the scratch CNN for each class by far.

4.3 Algorithm

The algorithm results are as expected. The algorithm is able to detect a dog breed, a human or something that is neither a dog nor a human.

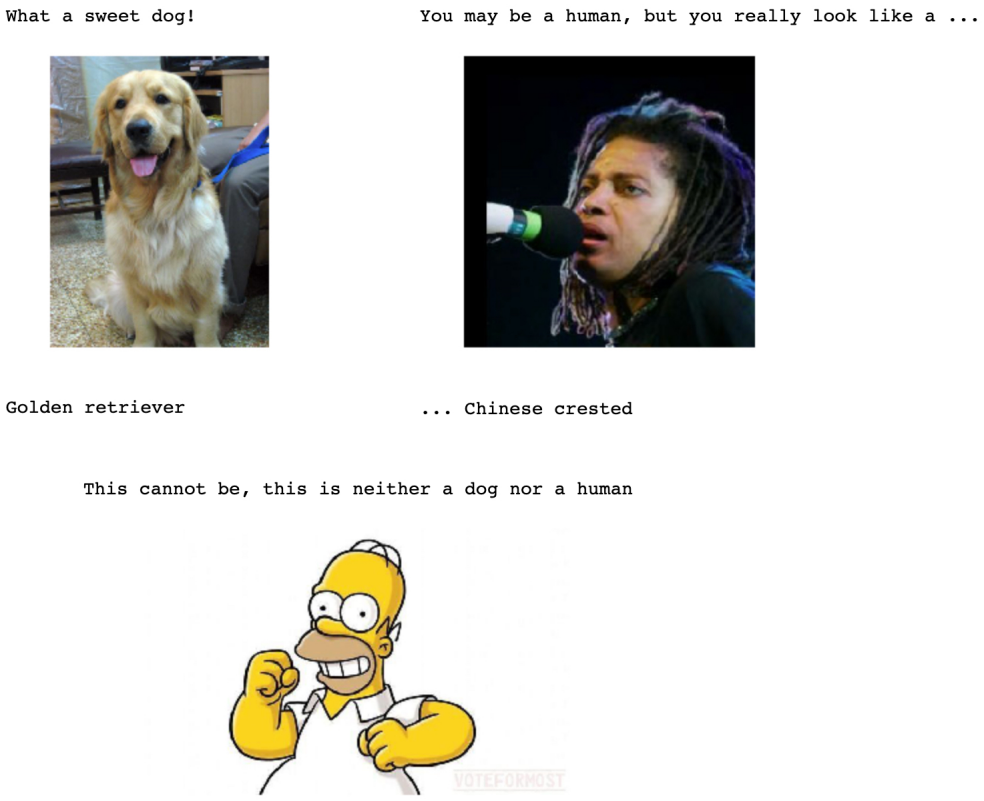

Figure 10: Correct algorithms results.

Since the model only reaches an accuracy of 81% there are false predictions. In figure 11 you can see two false predictions. The first dog is a West Highland White Terrier and the second dog is a black Labrador Retriever. The reason, why the model was not able to detect the West Highland White Terrier breed is, because the dog breed is not in the dataset. The model then returned the breed (Havanese) with the highest probability.

Figure 11: False dog breed predictions.

Improvements In general the algorithm results are good. The algorithm fulfills all requirements and returns precise predictions. There are some points that can be improved.
First, the algorithm is not able to predict more than one breed at a time. If the input is a group of at least two dogs, the algorithm will return only one prediction. It is also not clear which dog the algorithm will choose.
Secondly, if the input is a couple of a dog and a human, the algorithm returns only the dog breed prediction. A better result would be the dog breed prediction and the dog breed prediction for the human.

Figure 12: Example outputs for improvements.

References

[1] Cascade classifier. https://docs.opencv.org/.

[2] Labeled face in the wild. http://vis-www.cs.umass.edu/lfw.

[3] Performance metrics for classification problems in machine learning. https://medium.com/thalus-ai/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b.

[4] Pytorch. https://pytorch.org/.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. pages 770–778, June 2016.

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[7] Tibor Trnovszky, Patrik Kamencay, Richard Orješek, Miroslav Benco, and Peter Sykora. Animal recognition system based on convolutional neural network. *Advances in Electrical and Electronic Engineering*, 15, 10 2017.

[8] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. pages 511–518, 2001.